# Evaluation of ISO/IEC 9798 Protocols

# Version 2.0

David Basin and Cas Cremers

April 7, 2011

**Abstract**

This report provides a security evaluation of the authentication protocol families described in parts 2, 3, and 4 of the ISO-IEC 9798 standard. Our analysis includes formal models of the protocols and their security properties, an analysis of existing attacks and evaluations, a security analysis of the formal protocol models, and a list of recommendations.

## 1 Introduction

### 1.1 Scope

This report provides a security evaluation of the protocol families defined in the ISO-IEC standards 9798-2, 9898-3, and 9798-4. These three standards define 17 related protocol templates for entity authentication using symmetric cryptography, digital signatures, and cryptographic checksums respectively.

We have evaluated each protocol family as follows. First, we collected existing technical documents that evaluate the target protocol, checking the correctness of the reports. We then carried out a formal analysis using the Scyther tool [9]. We describe this tool in more detail in Section 3.1. Note that the protocols used apply cryptographic primitives in the "CRYPTREC recommended cipher list". Hence we apply symbolic analysis methods, without computational security.

Our analysis of the protocols encompasses the following properties.

1. The correct entity is authenticated,
2. The entity who has no right to authenticated is rejected,
3. Two sessions are not mistaken,
4. Resistance against other known attacks.

We have also commented on ambiguities and other unclear aspects, where applicable.

## 1.2 Reference documents

The ISO/IEC standards 9798-2, 9798-3, and 9798-4 define a suite of authentication protocols for different settings. We also take into account three technical corrigenda and one amendment.

1. ISO/IEC 9798-1:2010 [17]
2. ISO/IEC 9798-2:2008 [14]
3. ISO/IEC 9798-3:1998 [12]
4. ISO/IEC 9798-4:1999 [13]
5. ISO/IEC 9798-2:2008/Cor 1:2010 [18]
6. ISO/IEC 9798-3:1998/Cor 1:2009 [15]
7. ISO/IEC 9798-3:1998/Amd 1:2010 [19]
8. ISO/IEC 9798-4:1999/Cor 1:2009 [16]

## 1.3 Protocols

Each of the three standards considered defines authentication protocols based on different cryptographic mechanisms.

- ISO/IEC 9798-2 defines 6 protocols based on symmetric-key cryptography.
- ISO/IEC 9798-3 defines 7 protocols based on cryptographic signatures, of which five were present in the original standard ISO/IEC 9798-3:1998, and two that were introduced in the amendment Amd 1:2010.
- ISO/IEC 9798-4 defines 4 protocols based on cryptographic check functions.

In Tables 1–3 we provide an overview of the protocols. For each protocol, there are unilateral and bilateral authentication variants. The number of messages and passes differs among the protocols as well as the communication structure. Some of the protocols also use a trusted third party (TTP). Additionally, the protocols differ in the exact authentication properties provided.

Note that there is no consistent protocol naming scheme shared by the ISO documents. The symmetric-key based protocols are referred to in [14] as "mechanism 1", "mechanism 2", etc., whereas the protocols in [12, 16, 19] are referred to by their informal name, e. g., "One-pass unilateral authentication". In this document we will refer to protocols consistently by combining the document identifier, e. g., "9798-2" with a number $n$ to identify the $n$-th protocol in that document. For protocols that are proposed in an amendment, we continue the numbering from the base document. Hence we refer to the first protocol in [19] as "9798-3-6". The resulting unique identifiers are listed in the first column of the Tables 1–3.

| Protocol | Source | Mutual | TTP | Msgs | Name |
|---|---|---|---|---|---|
| 9798-2-1 | [14, p. 5] | N | N | 1 | One-pass unilateral authentication |
| 9798-2-2 | [14, p. 5] | N | N | 2 | Two-pass unilateral authentication |
| 9798-2-3 | [14, p. 6] | Y | N | 2 | Two-pass mutual authentication |
| 9798-2-4 | [14, p. 7] | Y | N | 3 | Three-pass mutual authentication |
| 9798-2-5 | [14, p. 8] | Y | Y | 4 | Four-pass with TTP |
| 9798-2-6 | [14, p. 10] | Y | Y | 5 | Five-pass with TTP |

Table 1: ISO/IEC 9798-2: Protocols based on symmetric cryptography [14]

| Protocol | Source | Mutual | TTP | Msgs | Name |
|---|---|---|---|---|---|
| 9798-3-1 | [12, p. 2] | N | N | 1 | One-pass unilateral authentication |
| 9798-3-2 | [12, p. 2] | N | N | 2 | Two-pass unilateral authentication |
| 9798-3-3 | [12, p. 3] | Y | N | 2 | Two-pass mutual authentication |
| 9798-3-4 | [12, p. 4] | Y | N | 3 | Three-pass mutual authentication |
| 9798-3-5 | [12, p. 4] | Y | N | 4 | Two-pass parallel mutual authentication |
| 9798-3-6 | [19, p. 2] | Y | Y | 5 | Five-pass mutual authentication with TTP, initiated by A |
| 9798-3-7 | [19, p. 4] | Y | Y | 5 | Five-pass mutual authentication with TTP, initiated by B |

Table 2: ISO/IEC 9798-3: Protocols based on cryptographic signatures [12,19]

| Protocol | Source | Mutual | TTP | Msgs | Name |
|---|---|---|---|---|---|
| 9798-4-1 | [13, p. 2] | N | N | 1 | One-pass unilateral authentication |
| 9798-4-2 | [13, p. 3] | N | N | 2 | Two-pass unilateral authentication |
| 9798-4-3 | [13, p. 4] | Y | N | 2 | Two-pass mutual authentication |
| 9798-4-4 | [13, p. 5] | Y | N | 3 | Three-pass mutual authentication |

Table 3: ISO/IEC 9798-4: Protocols based on cryptographic check functions [13]

Furthermore, most of the protocols are also parameterized by any combination of the following elements:

- All text fields included in the protocol specification are optional and their purpose is application-dependent.

- Many fields used to ensure uniqueness or freshness may be implemented either by random numbers, sequence numbers, or timestamps.
- Some protocols specify alternative message contents.
- Some identifier fields may be dropped, depending on implementation details.

### 1.4 Threat model and security properties

The ISO/IEC standards neither explicitly specify the threat model nor define any of the security properties that the protocols should satisfy. Instead, the introduction of ISO/IEC 9798-1 informally describes that the protocols should satisfy mutual or unilateral authentication. Furthermore, the following attacks are mentioned as being relevant:

1. man-in-the-middle attacks
2. replay attacks
3. reflection attacks
4. forced delay attacks

However, we note that the standards do not claim that any of the protocols are resilient against the above attacks.

### 1.5 Acknowledgements

Simon Meier assisted in repairing the protocols as well as the verification of the correctness of our repaired versions, which lead to significant improvements of our amendment proposals.

## 2 Known attacks

Some attacks based on parsing ambiguities [6] were reported on the original standards. These attacks lead to the corrigenda considered here and as a result, they do not occur on the corrected standards.

In [11], attacks on six protocols from the ISO/IEC 9798 standards are mentioned. Additional attacks on one of these six protocols are reported in [2]. These previously reported attacks form a strict subset of the attacks discovered by our analysis.

# 3 Analysis

## 3.1 Scyther

There are numerous robust tools available for formal security protocol analysis such as OFMC [1], Scyther [9], and ProVerif [5]. We have picked Scyther for the following reasons.

- It is very efficient at finding errors in protocols, i. e., generating counter examples to claimed properties.
- It can often succeed in verification, even for an unbounded number of protocol sessions.
- It provides support for modeling adversaries with different capabilities, in particular with respect to their ability to selectively compromise parts of participants' states, e. g., gaining access to their (long-term or short-term) secrets [3, 4].
- It is publicly available for download at [7].

The Scyther tool can perform bounded and unbounded verification. For details on the protocol model and the model checking algorithm underlying the Scyther tool we refer to [10].

## 3.2 Modeling protocols and their security properties

For our analysis, we modeled all 17 basic protocols plus 10 variants, resulting in 27 protocol models. In order to make our results reproducible, we give all protocol models in Appendix B.

In our protocol models, we have included the optional text fields and have omitted any checks on their contents on the receiver's side. We have modeled all uniqueness mechanisms as Nonces, that is, as fresh random numbers. We have modeled both versions of the protocols 9798-3-6 and 9798-3-7. For protocols whose identifier fields may be dropped under certain assumptions, we have also modeled their identifier-less variants with unidirectional keys.

In the Scyther framework, security properties are modeled by so-called *claims*. These can be viewed as local assertions made by agents: if they execute a protocol role up to a claim event, the claimed property is expected to hold. For example, if the final event of the $A$ role of a protocol is a claim of aliveness, then the property holds if and only if each time an honest agent executes the $A$ role, supposedly with peer $b$, who is honest, then $b$ previously should have performed an event (and hence is "alive").

For each protocol, we have modeled the following properties:

- Aliveness of the intended peer.
- Weak agreement with the peer on the involved agents and their roles.

- Data agreement with the peer on nonces and text fields, where possible.

Formal definitions of the properties are given in [8, 20]. In Table 4 we list the properties modeled for each protocol.

Protocols that satisfy the modeled agreement properties are resilient against man-in-the-middle attacks and reflection attacks. We have not explicitly modeled injective agreement, which implies resilience against replay attacks, because this is currently not supported by the Scyther tool. However, for the protocols specified by the ISO 9798 standards, resilience against replay attacks follows directly from agreement on nonces, timestamps, or sequences numbers, based on the requirements described in ISO/IEC 9798-1:2010 [17], Annex B: "Time variant parameters".

Other test details that are not specific to the tests performed here, such as the underlying protocol execution model, adversary model, security property definitions, and verification procedure, are discussed in detail in [4, 8].

## 3.3 Test details

In our tests we used the following setups.

- We limit the maximum number of threads that are involved in the attacks considered to five.
- We set a time limit for the analysis of a single security claim (property) of a protocol to ten minutes.
- We have analyzed all protocol properties in the single-protocol setting with respect to a typed model (i. e., if an agent expects to receive a message of type "time stamp", he will not accept a Nonce or an encryption, and vice versa) as well as an untyped model.

## 3.4 Analysis results

Our analysis reveals that the majority of the protocols in the standard meet their specified security properties. However, we have found several previously unreported attacks on five protocols and two protocol variants. These attacks fall into three categories: role-mixup attacks, type flaw attacks, and reflection attacks. In all cases, when the verifier finishes his role of the protocol, the protocol has not been executed as expected, which can lead to the verifier proceeding on false assumptions about the state of the claimant.

### 3.4.1 Role-mixup attacks

Authentication properties come in many forms. One well-known property, known as *agreement* [20], requires that when Alice finishes her role apparently with Bob, then Alice and Bob not only agree on the exchanged data, but additionally Alice can be sure that Bob was performing in the intended role. Some protocols are vulnerable to a *role-mixup attack*, in which an agent's assumptions

| | role A | | | role B | | |
|---|---|---|---|---|---|---|
| | Agreement | Aliveness | Weak agreement | Agreement | Aliveness | Weak agreement |
| 9798-2-1 | | | | {A,TNA,Text1} | • | • |
| 9798-2-1-udkey | | | | {A,TNA,Text1} | • | • |
| 9798-2-2 | | | | {A,RB,Text2} | • | • |
| 9798-2-2-udkey | | | | {A,RB,Text2} | • | • |
| 9798-2-3 | {B,TNB,Text3} | • | • | {A,TNA,Text1} | • | • |
| 9798-2-3-udkey | {B,TNB,Text3} | • | • | {A,TNA,Text1} | • | • |
| 9798-2-4 | {B,RA,RB,Text2,Text4} | • | • | {A,RA,RB,Text2} | • | • |
| 9798-2-4-udkey | {B,RA,RB,Text2,Text4} | • | • | {A,RA,RB,Text2} | • | • |
| 9798-2-5 | {B,Kab,Text5,Text7} | • | • | {A,Kab,Text5} | • | • |
| 9798-2-6 | {B,Kab,Text6,Text8} | • | • | {A,Kab,Text6} | • | • |
| 9798-3-1 | | | | {A,TNA,Text1} | • | • |
| 9798-3-2 | | | | {A,Ra,Rb,Text2} | • | • |
| 9798-3-3 | {B,TNB,Text3} | • | • | {A,TNA,Text1} | • | • |
| 9798-3-4 | {B,RA,RB,Text2,Text4} | • | • | {A,RA,RB,Text2} | • | • |
| 9798-3-5 | {B,RA,RB,Text5} | • | • | {A,RA,RB,Text3,Text5} | • | • |
| 9798-3-6-1 | {B,Ra,Rb,Text2} | • | | {A,Ra,Rb,Text8} | • | |
| 9798-3-6-2 | {B,Ra,Rb,Text2} | • | | {A,Ra,Rb,Text8} | • | |
| 9798-3-7-1 | {B,Ra,Rb,Text8} | • | | {A,Ra,Rb,Text6} | • | |
| 9798-3-7-2 | {B,Ra,Rb,Text8} | • | | {A,Ra,Rb,Text6} | • | |
| 9798-4-1 | | | | {A,TNA,Text1} | • | • |
| 9798-4-1-udkey | | | | {A,TNA,Text1} | • | • |
| 9798-4-2 | | | | {A,Rb,Text2} | • | • |
| 9798-4-2-udkey | | | | {A,Rb,Text2} | • | • |
| 9798-4-3 | {B,TNb,Text3} | • | • | {A,TNa,Text1} | • | • |
| 9798-4-3-udkey | {B,TNb,Text3} | • | • | {A,TNa,Text1} | • | • |
| 9798-4-4 | {B,Ra,Rb,Text2,Text4} | • | • | {A,Ra,Rb,Text2} | • | • |
| 9798-4-4-udkey | {B,Ra,Rb,Text2,Text4} | • | • | {A,Ra,Rb,Text2} | • | • |

Table 4: Overview of modeled security properties

on the role another agent is performing are wrong. Hence protocols vulnerable to such an attack do not satisfy matching conversations or synchronisation.

Figure 1 shows an example of a role-mixup attack on ISO/IEC 9798-2-3 with unidirectional keys. In the figure, boxes represent agent actions. Actions are executed within threads, represented by straight vertical lines. The top-most box of each thread denotes the parameters involved in the creation of the thread. Claims of security properties are denoted by hexagons. In this

figure, the hexagon is crossed out, denoting that the claimed property "weak agreement" does not hold. The horizontal arrows denote message flows.

In this attack, the adversary uses a message from Alice in the A role (Thread 1) to trick Alice into thinking that Bob is executing role A and is trying to initiate a session with her. However, Bob is only replying to a message provided to him by the adversary, and is executing role B.

An adversary can use such attacks, for example, to put Alice into a situation where she thinks that Bob has initiated a session with her, where in fact the adversary has triggered a responder session of Bob. The adversary thereby tricks Alice into thinking that Bob is in a different state than he actually is.

Additionally, in the case that the optional text fields Text1 and Text3 are used, the role-mixup attack can also violate the agreement property with respect to these fields: Alice will end the protocol under the assumption that the optional field data she receives from Bob was intended as Text1, whereas in fact Bob sent this data in the Text3 field. Depending on the exact use of the optional fields, this can constitute a serious security problem. Note that exploiting these attacks, as well as the other attacks described below, does not require "breaking" cryptography. Rather, the adversary exploits similarities among messages and the willingness of agents to engage in the protocol.

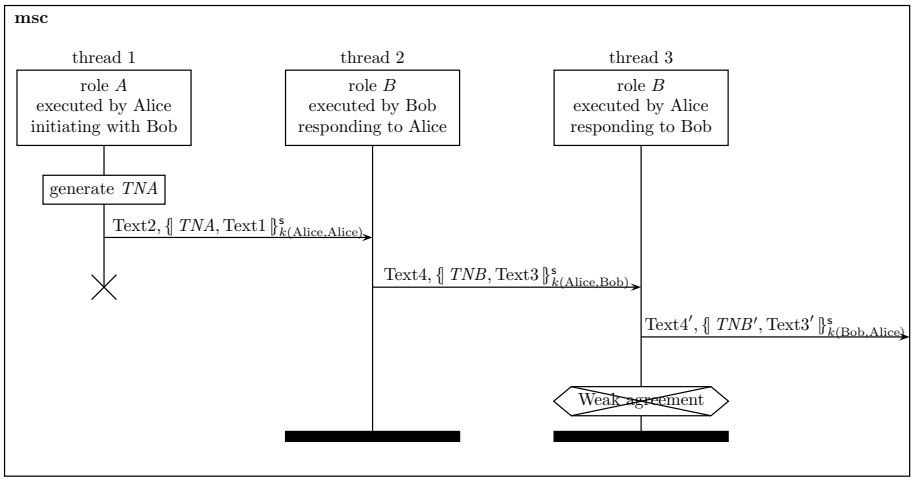


Figure 1: Role-mixup attack on ISO/IEC 9798-2-3 with unidirectional keys

Summarizing, we found role-mixup attacks on the following protocols.

1. ISO/IEC 9798-2-3 with bidirectional or unidirectional keys
2. ISO/IEC 9798-2-5
3. ISO/IEC 9798-3-3
4. ISO/IEC 9798-4-3 with bidirectional or unidirectional keys

### 3.4.2 Type flaw attacks

Some implementations are vulnerable to so-called *type flaw attacks*. Consider, for example, an implementation in which agent names are encoded into bit-fields of length $n$, which is also the length of bit-fields representing nonces. It may then happen that an agent expects to receive a nonce (any fresh random value which it has not seen before), and may therefore accept a bit string that was intended to represent an agent name.

Our automatic analysis has revealed such an attack on the ISO/IEC 9897-3-7 protocol, also referred to as "Five pass authentication (initiated by B)" [19, p. 4], where the tokens are interpreted according to "Option 1" in the original specification. In the attack, both (agent) Alice and (trusted party) Terence mistakenly accept the bit string corresponding to the agent name "Alice" as the value of a nonce.

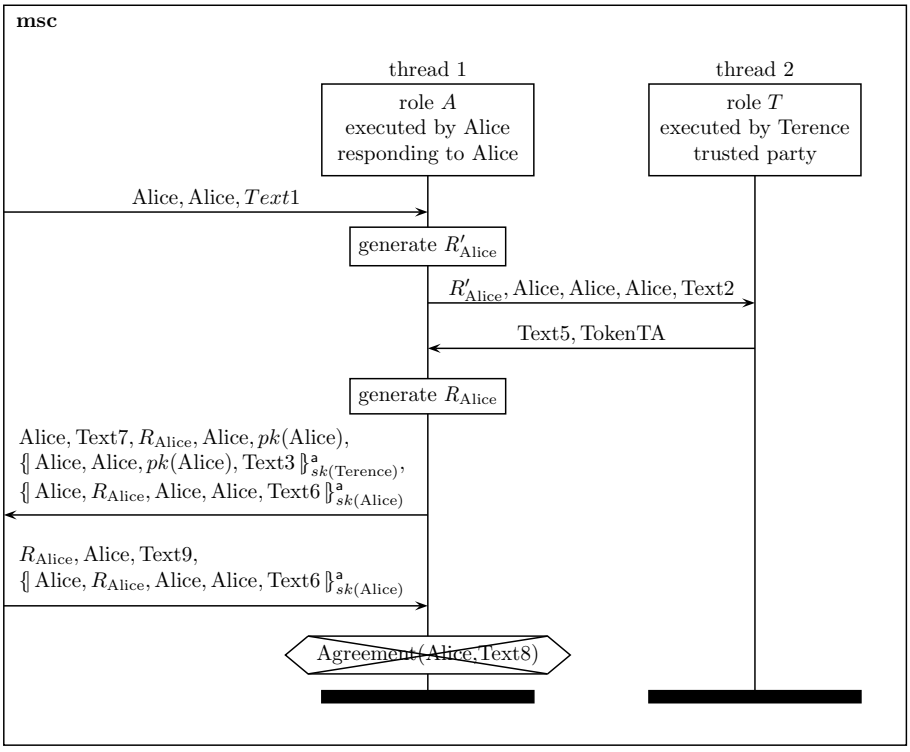


Figure 2: Type flaw attack on ISO/IEC 9798-3-7 (Option 1)

Note that the attack does not require an agent to start a communication with himself: Alice responds to an incoming message and, following the standard, accepts any agent identity as the sender.

The attack also works when optional fields are used; however, it is then

required that the variable for the Text8 field can be instantiated with the value of the Text6 field. The attack exploits the fact that when the adversary can induce $R_B = A = B$, then the second signature in TokenAB coincides with the signature in TokenBA, even though Alice checks the value of $R_A$ (which is in both signatures in the second position).

Summarizing, we found type flaw attacks on the following protocol.

1. ISO/IEC 9798-3-7 (Option 1)

### 3.4.3 Reflection attacks

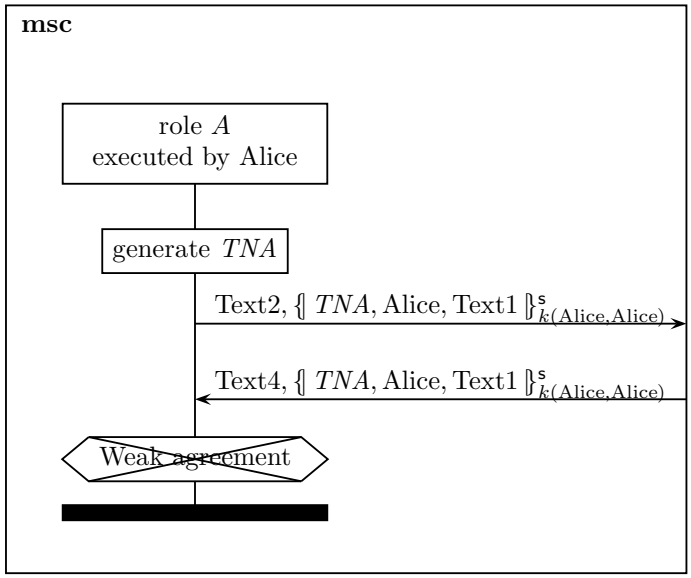


Figure 3: Alice-talks-to-Alice reflection attack on ISO/IEC 9798-2-3

The reflection attacks we found on the ISO/IEC 9798 protocols occur only when agents may start sessions communicating with the same identity, a so-called *Alice-talks-to-Alice* scenario. The feasibility and relevance of this scenario depends on the application and its internal checks.

If an Alice-talks-to-Alice scenario is possible, some protocols are vulnerable to a reflection attack. The Message Sequence Chart in Figure 3 shows an example of an Alice-talks-to-Alice reflection attack on ISO/IEC 9798-2-3. In this attack, the adversary (not depicted) reflects the encrypted part of the message sent by Alice back to the same thread, while prepending the message Text4. Notice that Alice expects Text3 in the encrypted message she receives. As the purpose of this optional field is not specified by the standard, the attack works if (a) the optional fields Text1 and Text3 are omitted, or (b) the checks that

Alice performs on Text3 (e. g., check that it is a string) can be passed by Text1. The adversary is free to insert any possible Text4.

The attack violates one of the main requirements stated in the ISO/IEC 9798-1 introduction, namely absence of "reflection attacks".

Summarizing, we found Alice-talks-to-Alice reflection attacks on the following protocols.

1. ISO/IEC 9798-2-3
2. ISO/IEC 9798-2-5
3. ISO/IEC 9798-4-3

## 3.5 Attack overview

In Table 5 we list the attacks we found using the Scyther tool. The rows list the protocols and their violated properties. A dot in the table indicates an attack under the conditions mentioned in the column headers.

## 3.6 Fixing the problems

**Tagging** The majority of the above attacks can be mitigated by following prudent engineering practices for security protocols. In particular, *tagging* would prevent most of the attacks: include globally unique constants in each cryptographically protected message, to ensure that components from one part of the protocol cannot be misinterpreted as components from another part of the same protocol or some other protocol variant.

However, tagging alone is not sufficient to prevent all weaknesses in the standards. There are two remaining issues: ambiguity of optional fields in multi-protocol settings, and unstated assumptions for the ISO/IEC 9798-2-5 and ISO/IEC 9798-2-6 protocols.

**Removing ambiguity of optional fields** Though not explored in-depth in this report, the interaction between various protocol variants can lead to violation of protocol properties even when protocols are tagged. In particular, some protocols contain a sequence of the following form: an optional identity field followed by an optional payload field.

For example, consider a protocol in which a message contains the following sequence: $X||ID_A||Text$, where $ID_A$ is an identity field that may be dropped (e. g., when unidirectional keys are used) and *Text* is an optional text field. Then, it may be the case that in one variant of the protocol, an agent expects a message of the form $X||ID_A$, whereas the other implementation expects a message of the form $X||Text$. The interaction between the two interpretations can enable attacks, for example, when the text field is used to insert a false agent identity, or where an agent identity is wrongly assumed to be the content of the text field.

| No | Protocol | Claim | No type checks Alice-talks-to-Alice initators | Type checks Alice-talks-to-Alice initators | No type checks No Alice-talks-to-Alice initators | Type checks No Alice-talks-to-Alice initators |
|---|---|---|---|---|---|---|
| 1 | isoiec-9798-2-3 | A Agreement(B,TNB,Text3) | • | • | • | • |
| 2 | isoiec-9798-2-3 | A Weakagree | • | • | • | • |
| 3 | isoiec-9798-2-3 | B Agreement(A,TNA,Text1) | • | • | • | • |
| 4 | isoiec-9798-2-3-udkey | A Agreement(B,TNB,Text3) | • | • | • | • |
| 5 | isoiec-9798-2-3-udkey | A Weakagree | • | • | • | • |
| 6 | isoiec-9798-2-3-udkey | B Agreement(A,TNA,Text1) | • | • | • | • |
| 7 | isoiec-9798-2-5 | A Alive | • | • | | |
| 8 | isoiec-9798-2-5 | A Agreement(B,Kab,Text5,Text7) | • | • | | |
| 9 | isoiec-9798-2-5 | A Weakagree | • | • | | |
| 10 | isoiec-9798-2-5 | B Alive | • | • | • | • |
| 11 | isoiec-9798-2-5 | B Agreement(A,Kab,Text5) | • | • | • | • |
| 12 | isoiec-9798-2-6 | A Alive | • | • | • | • |
| 13 | isoiec-9798-2-6 | A Agreement(B,Kab,Text6,Text8) | • | • | • | • |
| 14 | isoiec-9798-2-6 | B Alive | • | • | • | • |
| 15 | isoiec-9798-2-6 | B Agreement(A,Kab,Text6) | • | • | • | • |
| 16 | isoiec-9798-3-3 | A Agreement(B,TNB,Text3) | • | • | • | • |
| 17 | isoiec-9798-3-3 | A Weakagree | • | • | • | • |
| 18 | isoiec-9798-3-3 | B Agreement(A,TNA,Text1) | • | • | • | • |
| 19 | isoiec-9798-3-7-1 | A Agreement(B,Ra,Rb,Text8) | • | | • | |
| 20 | isoiec-9798-4-3 | A Agreement(B,TNb,Text3) | • | • | • | • |
| 21 | isoiec-9798-4-3 | A Weakagree | • | • | • | • |
| 22 | isoiec-9798-4-3 | B Agreement(A,TNa,Text1) | • | • | • | • |
| 23 | isoiec-9798-4-3-udkey | A Agreement(B,TNb,Text3) | • | • | • | • |
| 24 | isoiec-9798-4-3-udkey | A Weakagree | • | • | • | • |
| 25 | isoiec-9798-4-3-udkey | B Agreement(A,TNa,Text1) | • | • | • | • |

Table 5: Attack overview

The ambiguity can be resolved by implementing optional fields by a variable-length field. When the optional field is empty, the length is set to zero. In such an implementation the above expected messages correspond to $X||ID_A|| \perp$ and $X|| \perp ||Text$, respectively, where $\perp$ denotes the zero-length field.

**Explicitly stating assumptions on ISO/IEC 9798-2-5 and 9798-2-6** The signature-based protocols with a TTP, ISO/IEC 9798-2-5 and 9798-2-6, do not explicitly state that entities performing the TTP role cannot perform other roles. The specification therefore allows that Alice may perform both the role of the TTP as well as role A or B. In that case, there is a significant attack on the protocol, and aliveness of the partner cannot be guaranteed.

# 4 Recommendations

## Critical

Address the protocol attacks reported here. This may be done by either (a) removing the affected protocols from the standards, (b) updating their description to reflect the constraints under which they work, or (c) fixing the attacks by including appropriate tagging and further minor modifications as described in Section 3.6. We strongly recommend solution (c) and we provide proposals for amending the standards in Appendix A.

## Medium

The threat model and security properties should be made explicit in order to clarify what is gained by using the protocols.

## Low

Readers of the standard may not know how to choose among the different protocols. The standard could provide more guidance with respect to the consequences of particular choices.

In many protocols, some text fields are unprotected, i. e., an adversary can arbitrarily change them, whereas other text fields are authenticated or secret. Making these differences explicit would enhance the usability of the standards.

## Editorial

The naming of the protocols is not consistent among the 9798 standards. The standard 9798-2 explicitly enumerates its protocols as "mechanism 1" to "mechanism 6". This is not the case for 9798-3 or 9798-4, in which protocols are often referred to by their informal names, for example "one pass authentication" is used to refer to the first protocol of 9798-3. We recommend the use of a numbering scheme similar to the one used in ISO/IEC 9798-2.

# References


[1] D. B. Sebastian Mödersheim Luca Viganò. Ofmc: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, June 2005. Published online December 2004.

[2] A. Armando and L. Compagna. SAT-based model-checking for security protocols analysis. *Int. J. Inf. Sec.*, 7(1):3–32, 2008.

[3] D. Basin and C. Cremers. Degrees of security: Protocol guarantees in the face of compromising adversaries. In A. Dawar and H. Veith, editors, *19th EACSL Annual Conference on Computer Science Logic (CSL)*, volume 6247 of *LNCS*, pages 1–18, Brno, Czech Republic, 08 2010. Springer-Verlag.

[4] D. Basin and C. Cremers. Modeling and analyzing security in the presence of compromising adversaries. In *Computer Security - ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 340–356. Springer, 2010.

[5] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE, 2001.

[6] L. Chen and C. J. Mitchell. Parsing ambiguities in authentication and key establishment protocols. *Int. J. Electron. Secur. Digit. Forensic*, 3:82–94, March 2010.

[7] C. Cremers. Scyther tool with compromising adversaries extension. Includes protocol description files and test scripts. Available online at `http://people.inf.ethz.ch/cremersc/scyther/`.

[8] C. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.

[9] C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proc. CAV*, volume 5123 of *LNCS*, pages 414–418. Springer, 2008.

[10] C. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *CCS '08: Proc. of the 15th ACM conference on Computer and communications security*, pages 119–128. ACM, 2008.

[11] B. Donovan, P. Norris, and G. Lowe. Analyzing a library of security protocols using Casper and FDR. In *Proceedings of the Workshop on Formal Methods and Security Protocols*, 1999. Some of the Casper scripts are available here: `http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Security/Papers/prots.tar.gz`.

[12] International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-3:1998, Information technology – Security techniques – Entity Authentication – Part 3: Mechanisms using digital signature techniques, 1998. Second edition.

[13] International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-4:1999, Information technology – Security techniques – Entity Authentication – Part 3: Mechanisms using a cryptographic check function, 1999. Second edition.

[14] International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-2:2008, Information technology – Security techniques – Entity Authentication – Part 2: Mechanisms using symmetric encipherment algorithms, 2008. Third edition.

[15] International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-3:1998/Cor.1:2009, Information technology – Security techniques – Entity Authentication – Part 3: Mechanisms using digital signature techniques. Technical Corrigendum 1, 2009.

[16] International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-4:1999/Cor.1:2009, Information technology – Security techniques – Entity Authentication – Part 3: Mechanisms using a cryptographic check function. Technical Corrigendum 1, 2009.

[17] International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-1:2010, Information technology – Security techniques – Entity Authentication – Part 1: General, 2010. Third edition.

[18] International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-2:2008/Cor.1:2010, Information technology – Security techniques – Entity Authentication – Part 2: Mechanisms using symmetric encipherment algorithms. Technical Corrigendum 1, 2010.

[19] International Organization for Standardization, Genève, Switzerland. ISO/IEC 9798-3:1998/Amd.1:2010, Information technology – Security techniques – Entity Authentication – Part 3: Mechanisms using digital signature techniques. Amendment 1, 2010.

[20] G. Lowe. A hierarchy of authentication specifications. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 31–44. IEEE, 1997.

# A Suggested amendments to the ISO/IEC 9798 standards

## A.1 ISO/IEC 9798-1:2010 Technical Corrigendum

Page 10, before bibliography.
Insert the following section:

**Annex D**
(informative)

**Optional Fields**

When optional fields, such as optional identities or optional text fields, are not used then they must be set to empty. In particular, the encoding of the messages must ensure that the concatenation of a field and an empty optional field is uniquely parsed as a concatenation. This can be achieved by implementing optional fields as variable-length fields. If the optional field is not used, the length of the field is set to zero.

## A.2 ISO/IEC 9798-2:2008 Technical Corrigendum 2

Page 2, clause 7
Insert the following text after this clause:

> The encrypted data used at different places in the protocols must not be interchangeable. This may be enforced by including in each encryption the following two elements:
>
> 1. The object identifier as specified in Annex B [19, p. 6], in particular identifying the ISO standard, the part number, and the authentication mechanism.
> 2. For protocols that contain more than one encryption, each encryption must contain a constant that uniquely identifies the position of the encryption within the protocol.
>
> The recipient of an encrypted message must verify that the object identifier and the encryption position identifier are as expected.
> The encryption keys used by implementations of the ISO/IEC 9798-2 protocols must be distinct from the keys used by other protocols.

Page 3, Section 7
Insert the following text before Section 7.1:

> Entities that perform the role of the TTP must not perform the roles A and B.

## A.3 ISO/IEC 9798-3:1998 Technical Corrigendum 2

Page 2, clause 7
Insert the following text after this clause:

> The signed data used at different places in the protocols must not be interchangeable. This may be enforced by including in each signature the following two elements:
>
> 1. The object identifier as specified in Annex B [19, p. 6], in particular identifying the ISO standard, the part number, and the authentication mechanism.
> 2. For protocols that contain more than one signature, each signature must contain a constant that uniquely identifies the position of the signature within the protocol.
>
> The recipient of a signature must verify that the object identifier and the signature position identifier are as expected.
> The signature keys used by implementations of the ISO/IEC 9798-3 protocols must be distinct from the keys used by other protocols.

### A.4 ISO/IEC 9798-4:1999 Technical Corrigendum 2

Page 2, Section 4
Insert the following text at the end of Section 4:

> The cryptographic check data used at different places in the protocols must not be interchangeable. This may be enforced by including in each cryptographic check the following two elements:
>
> 1. The object identifier as specified in Annex B [19, p. 6], in particular identifying the ISO standard, the part number, and the authentication mechanism.
> 2. For protocols that contain more than one cryptographic check, each cryptographic check must contain a constant that uniquely identifies the position of the cryptographic check within the protocol.
>
> The recipient of a cryptographic check must verify that the object identifier and the cryptographic check position identifier are as expected.
> The keys used by implementations of the ISO/IEC 9798-4 protocols must be distinct from the keys used by other protocols.

## B Protocol models

In this Appendix, we provide the complete source files as they were provided to the Scyther tool, Compromise version 0.6. The tool can be downloaded from `http://people.inf.ethz.ch/cremersc/scyther/compromise/index.html`.

### B.1 ISO/IEC 9798-2 protocols

Listing 1: ISO/IEC 9798-2-1

```
1  /*
2   * Modeled from ISO/IEC 9798
3   * Modeler: Cas Cremers, Dec. 2010
4   *
5   * symmetric
6   * one-pass
7   * unilateral
8   *
9   * Note: the identity B may be ommitted, if
10  *   (a) the environment disallows such attacks, or
11  *   (b) a unidirectional key is used
12  */
13 protocol @keysymm-21(A,B)
14 {
15    role A
16    {
17       var T: Nonce;
18       var Text: Ticket;
19
```

```
20        recv_!1(B,A, { T, A, Text }k(A,B) );
21        send_!2(A,B, { T, A, Text }k(B,A) );
22     }
23     role B
24     {
25        var T: Nonce;
26        var Text: Ticket;
27
28        recv_!3(A,B, { T, B, Text }k(A,B) );
29        send_!4(B,A, { T, B, Text }k(B,A) );
30     }
31  }
32
33  protocol isoiec-9798-2-1(A,B)
34  {
35     role A
36     {
37        fresh TNA: Nonce;
38        fresh Text1,Text2: Ticket;
39
40        claim(A,Running,B,TNA,Text1);
41        send_1(A,B, Text2, { TNA, B, Text1 }k(A,B) );
42     }
43     role B
44     {
45        var TNA: Nonce;
46        var Text1,Text2: Ticket;
47
48        recv_1(A,B, Text2, { TNA, B, Text1 }k(A,B) );
49
50        claim(B,Commit,A,TNA,Text1);
51        claim(B,Alive);
52        claim(B,Weakagree);
53     }
54  }
```

Listing 2: ISO/IEC 9798-2-1 with unidirectional keys

```
1  /*
2   * Modeled from ISO/IEC 9798
3   * Modeler: Cas Cremers, Dec. 2010
4   *
5   * symmetric
6   * one-pass
7   * unilateral
8   *
9   * Note: the identity B may be ommitted, if
10  *   (a) the environment disallows such attacks, or
11  *   (b) a unidirectional key is used
12  */
13 protocol isoiec-9798-2-1-udkey(A,B)
14 {
15    role A
16    {
17       fresh TNA: Nonce;
18       fresh Text1,Text2: Ticket;
19
```

```
20        claim(A,Running,B,TNA,Text1);
21        send_1(A,B, Text2, { TNA, Text1 }k(A,B) );
22     }
23     role B
24     {
25        var TNA: Nonce;
26        var Text1,Text2: Ticket;
27
28        recv_1(A,B, Text2, { TNA, Text1 }k(A,B) );
29
30        claim(B,Commit,A,TNA,Text1);
31        claim(B,Alive);
32        claim(B,Weakagree);
33     }
34  }
```

Listing 3: ISO/IEC 9798-2-2

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * symmetric
6    * two-pass
7    * unilateral
8    *
9    * Note: the identity A may be ommitted, if
10   *   (a) the environment disallows such attacks, or
11   *   (b) a unidirectional key is used
12   */
13  protocol @keysymm-22(A,B)
14  {
15     role A
16     {
17        var T: Nonce;
18        var Text: Ticket;
19
20        recv_!1(B,A, { T, A, Text }k(A,B) );
21        send_!2(A,B, { T, A, Text }k(B,A) );
22     }
23     role B
24     {
25        var T: Nonce;
26        var Text: Ticket;
27
28        recv_!3(A,B, { T, B, Text }k(A,B) );
29        send_!4(B,A, { T, B, Text }k(B,A) );
30     }
31  }
32
33  protocol isoiec-9798-2-2(A,B)
34  {
35     role A
36     {
37        var RB: Nonce;
38        var Text1: Ticket;
39        fresh Text2,Text3: Ticket;
```

```
40
41          recv_1(B,A, RB,Text1 );
42          claim(A,Running,B,RB,Text2);
43          send_2(A,B, Text3, { RB, B, Text2 }k(B,A) );
44       }
45       role B
46       {
47          fresh RB: Nonce;
48          fresh Text1: Ticket;
49          var Text2,Text3: Ticket;
50
51          send_1(B,A, RB,Text1 );
52          recv_2(A,B, Text3, { RB, B, Text2 }k(B,A) );
53
54          claim(B,Commit,A,RB,Text2);
55          claim(B,Alive);
56          claim(B,Weakagree);
57       }
58    }
```

Listing 4: ISO/IEC 9798-2-2 with unidirectional keys

```
1  /*
2   * Modeled from ISO/IEC 9798
3   * Modeler: Cas Cremers, Dec. 2010
4   *
5   * symmetric
6   * two-pass
7   * unilateral
8   *
9   * Note: the identity A may be ommitted, if
10  *   (a) the environment disallows such attacks, or
11  *   (b) a unidirectional key is used
12  *
13  */
14 protocol isoiec-9798-2-2-udkey(A,B)
15 {
16    role A
17    {
18       var RB: Nonce;
19       var Text1: Ticket;
20       fresh Text2,Text3: Ticket;
21
22       recv_1(B,A, RB,Text1 );
23       claim(A,Running,B,RB,Text2);
24       send_2(A,B, Text3, { RB, Text2 }k(B,A) );
25    }
26    role B
27    {
28       fresh RB: Nonce;
29       fresh Text1: Ticket;
30       var Text2,Text3: Ticket;
31
32       send_1(B,A, RB,Text1 );
33       recv_2(A,B, Text3, { RB, Text2 }k(B,A) );
34
35       claim(B,Commit,A,RB,Text2);
```

```
36        claim(B,Alive);
37        claim(B,Weakagree);
38     }
39  }
```

Listing 5: ISO/IEC 9798-2-3

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * symmetric
6    * two-pass
7    * mutual
8    *
9    * Note: the identity inside the encryption may be ommitted, if
10   *   (a) the environment disallows such attacks, or
11   *   (b) a unidirectional key is used
12   *
13   */
14  protocol @keysymm-23(A,B)
15  {
16     role A
17     {
18        var T: Nonce;
19        var Text: Ticket;
20
21        recv_!1(B,A, { T, A, Text }k(A,B) );
22        send_!2(A,B, { T, A, Text }k(B,A) );
23     }
24     role B
25     {
26        var T: Nonce;
27        var Text: Ticket;
28
29        recv_!3(A,B, { T, B, Text }k(A,B) );
30        send_!4(B,A, { T, B, Text }k(B,A) );
31     }
32  }
33
34  protocol isoiec-9798-2-3(A,B)
35  {
36     role A
37     {
38        fresh TNA: Nonce;
39        var TNB: Nonce;
40        fresh Text1,Text2: Ticket;
41        var Text3,Text4: Ticket;
42
43        claim(A,Running,B,TNA,Text1);
44        send_1(A,B, Text2, { TNA, B, Text1 }k(A,B) );
45        recv_2(B,A, Text4, { TNB, A, Text3 }k(A,B) );
46
47        claim(A,Commit,B,TNB,Text3);
48        claim(A,Alive);
49        claim(A,Weakagree);
50     }
```

```
51      role B
52      {
53          var TNA: Nonce;
54          fresh TNB: Nonce;
55          var Text1,Text2: Ticket;
56          fresh Text3,Text4: Ticket;
57
58          recv_1(A,B, Text2, { TNA, B, Text1 }k(A,B) );
59          claim(B,Running,A,TNB,Text3);
60          send_2(B,A, Text4, { TNB, A, Text3 }k(A,B) );
61
62          claim(B,Commit,A,TNA,Text1);
63          claim(B,Alive);
64          claim(B,Weakagree);
65      }
66  }
```

Listing 6: ISO/IEC 9798-2-3 with unidirectional keys

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * symmetric
6    * two-pass
7    * mutual
8    *
9    * Note: the identity inside the encryption may be ommitted, if
10   *   (a) the environment disallows such attacks, or
11   *   (b) a unidirectional key is used
12   *
13   * In case (b), modeled here, the second key is reversed.
14   *
15   */
16  protocol isoiec-9798-2-3-udkey(A,B)
17  {
18      role A
19      {
20          fresh TNA: Nonce;
21          var TNB: Nonce;
22          fresh Text1,Text2: Ticket;
23          var Text3,Text4: Ticket;
24
25          claim(A,Running,B,TNA,Text1);
26          send_1(A,B, Text2, { TNA, Text1 }k(A,B) );
27          recv_2(B,A, Text4, { TNB, Text3 }k(B,A) );
28
29          claim(A,Commit,B,TNB,Text3);
30          claim(A,Alive);
31          claim(A,Weakagree);
32      }
33      role B
34      {
35          var TNA: Nonce;
36          fresh TNB: Nonce;
37          var Text1,Text2: Ticket;
38          fresh Text3,Text4: Ticket;
```

```
39
40        recv_1(A,B, Text2, { TNA, Text1 }k(A,B) );
41        claim(B,Running,A,TNB,Text3);
42        send_2(B,A, Text4, { TNB, Text3 }k(B,A) );
43
44        claim(B,Commit,A,TNA,Text1);
45        claim(B,Alive);
46        claim(B,Weakagree);
47     }
48  }
```

Listing 7: ISO/IEC 9798-2-4

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * symmetric
6    * three-pass
7    * mutual
8    *
9    * Note: the identity inside the encryption may be ommitted, if
10   *   (a) the environment disallows such attacks, or
11   *   (b) a unidirectional key is used
12   */
13  protocol @keysymm-24a(A,B)
14  {
15     role A
16     {
17        var T1,T2: Nonce;
18        var Text: Ticket;
19
20        recv_!1(B,A, { T1, T2, A, Text }k(A,B) );
21        send_!2(A,B, { T1, T2, A, Text }k(B,A) );
22     }
23     role B
24     {
25        var T1,T2: Nonce;
26        var Text: Ticket;
27
28        recv_!3(A,B, { T1, T2, B, Text }k(A,B) );
29        send_!4(B,A, { T1, T2, B, Text }k(B,A) );
30     }
31  }
32
33  protocol @keysymm-24b(A,B)
34  {
35     role A
36     {
37        var T1,T2: Nonce;
38        var Text: Ticket;
39
40        recv_!1(B,A, { T1, T2, Text }k(A,B) );
41        send_!2(A,B, { T1, T2, Text }k(B,A) );
42     }
43     role B
44     {
```

```
45        var T1,T2: Nonce;
46        var Text: Ticket;
47
48        recv_!3(A,B, { T1, T2, Text }k(A,B) );
49        send_!4(B,A, { T1, T2, Text }k(B,A) );
50     }
51  }
52
53  protocol isoiec-9798-2-4(A,B)
54  {
55     role A
56     {
57        var RB: Nonce;
58        fresh RA: Nonce;
59        var Text1,Text4,Text5: Ticket;
60        fresh Text2,Text3: Ticket;
61
62        recv_1(B,A, RB,Text1 );
63        claim(A,Running,B,RA,RB,Text2);
64        send_2(A,B, Text3, { RA, RB, B, Text2 }k(A,B) );
65        recv_3(B,A, Text5, { RB, RA, Text4 }k(A,B) );
66
67        claim(A,Commit,B,RA,RB,Text2,Text4);
68        claim(A,Alive);
69        claim(A,Weakagree);
70     }
71     role B
72     {
73        fresh RB: Nonce;
74        var RA: Nonce;
75        fresh Text1,Text4,Text5: Ticket;
76        var Text2,Text3: Ticket;
77
78        send_1(B,A, RB,Text1 );
79        recv_2(A,B, Text3, { RA, RB, B, Text2 }k(A,B) );
80        claim(B,Running,A,RA,RB,Text2,Text4);
81        send_3(B,A, Text5, { RB, RA, Text4 }k(A,B) );
82
83        claim(B,Commit,A,RA,RB,Text2);
84        claim(B,Alive);
85        claim(B,Weakagree);
86     }
87  }
```

Listing 8: ISO/IEC 9798-2-4 with unidirectional keys

```
1  /*
2   * Modeled from ISO/IEC 9798
3   * Modeler: Cas Cremers, Dec. 2010
4   *
5   * symmetric
6   * three-pass
7   * mutual
8   *
9   * Note: the identity inside the encryption may be ommitted, if
10  *   (a) the environment disallows such attacks, or
11  *   (b) a unidirectional key is used
```

```
 *
 * In case (b), modeled here, the second key is reversed.
 */
protocol isoiec-9798-2-4-udkey(A,B)
{
   role A
   {
      var RB: Nonce;
      fresh RA: Nonce;
      var Text1,Text4,Text5: Ticket;
      fresh Text2,Text3: Ticket;

      recv_1(B,A, RB,Text1 );
      claim(A,Running,B,RA,RB,Text2);
      send_2(A,B, Text3, { RA, RB, Text2 }k(A,B) );
      recv_3(B,A, Text5, { RB, RA, Text4 }k(B,A) );

      claim(A,Commit,B,RA,RB,Text2,Text4);
      claim(A,Alive);
      claim(A,Weakagree);
   }
   role B
   {
      fresh RB: Nonce;
      var RA: Nonce;
      fresh Text1,Text4,Text5: Ticket;
      var Text2,Text3: Ticket;

      send_1(B,A, RB,Text1 );
      recv_2(A,B, Text3, { RA, RB, Text2 }k(A,B) );
      claim(B,Running,A,RA,RB,Text2,Text4);
      send_3(B,A, Text5, { RB, RA, Text4 }k(B,A) );

      claim(B,Commit,A,RA,RB,Text2);
      claim(B,Alive);
      claim(B,Weakagree);
   }
}
```

Listing 9: ISO/IEC 9798-2-5

```
/*
 * Modeled from ISO/IEC 9798
 * Modeler: Cas Cremers, Dec. 2010
 *
 * symmetric
 * ttp
 * four-pass
 * mutual
 *
 * Modeling notes:
 * - The use of TNb in message 4, as specified by the ISO standard, is
 *   different from other models, in which it was TNa.
 */
usertype SessionKey;

```

```
16 protocol isoiec-9798-2-5(A,B,P)
17 {
18    role A
19    {
20       fresh TVPa: Nonce;
21       var T: Ticket;
22       fresh TNa: Nonce;
23       var TNb: Nonce;
24       var Kab: SessionKey;
25       fresh Text1,Text5,Text6: Ticket;
26       var Text3,Text4,Text7,Text8: Ticket;
27
28       send_1(A,P, TVPa, B, Text1);
29       recv_2(P,A, Text4, { TVPa, Kab, B, Text3 }k(A,P), T );
30       claim(A,Running,B,Kab,Text5);
31       send_3(A,B, Text6, T, { TNa, B, Text5 }Kab );
32       recv_4(B,A, Text8, { TNb, A, Text7 }Kab );
33
34       claim(A,Commit,B,Kab,Text5,Text7);
35       claim(A,Secret,Kab);
36       claim(A,Secret,Text5);
37       claim(A,Secret,Text7);
38       claim(A,Alive);
39       claim(A,Weakagree);
40    }
41    role B
42    {
43       var TNp: Nonce;
44       var TNa: Nonce;
45       fresh TNb: Nonce;
46       var Kab: SessionKey;
47       fresh Text7,Text8: Ticket;
48       var Text2,Text5,Text6: Ticket;
49
50       recv_3(A,B, Text6, { TNp, Kab, A, Text2  }k(B,P), { TNa, B,
             Text5 }Kab );
51       claim(B,Running,A,Kab,Text5,Text7);
52       send_4(B,A, Text8, { TNb, A, Text7 }Kab );
53
54       claim(B,Commit,A,Kab,Text5);
55       claim(B,Secret,Kab);
56       claim(B,Secret,Text5);
57       claim(B,Secret,Text7);
58       claim(B,Alive);
59       claim(B,Weakagree);
60    }
61    role P
62    {
63       var TVPa: Nonce;
64       fresh TNp: Nonce;
65       fresh Kab: SessionKey;
66       fresh Text2,Text3,Text4: Ticket;
67       var Text1: Ticket;
68
69       recv_1(A,P, TVPa, B, Text1);
70       send_2(P,A, Text4, { TVPa, Kab, B, Text3 }k(A,P),
71              { TNp, Kab, A, Text2  }k(B,P) );
```

```
72      }
73  }
74
75  protocol @keysymm25(A,B,P)
76  {
77      role A
78      {
79          var TVPN: Nonce;
80          var Kab: SessionKey;
81          var Text: Ticket;
82
83          recv_!1(B,A, { TVPN, Kab, B, Text }k(P,A) );
84          send_!2(A,B, { TVPN, Kab, B, Text }k(A,P) );
85      }
86      role B
87      {
88      }
89      role P
90      {
91      }
92  }
```

Listing 10: ISO/IEC 9798-2-6

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * symmetric
6    * ttp
7    * five-pass
8    * mutual
9    *
10   * MPA Attack reported by Mathuria:
11   * - Type flaw MPA when in parallel with Abadi-Needham protocol.
12   *
13   */
14  protocol isoiec-9798-2-6(A,B,P)
15  {
16      role A
17      {
18          var Rb: Nonce;
19          fresh Ra,Rpa: Nonce;
20          var Kab: SessionKey;
21          var T: Ticket;
22          fresh Text2,Text6,Text7: Ticket;
23          var Text1,Text4,Text5,Text8,Text9: Ticket;
24
25          recv_1(B,A, Rb, Text1);
26          send_2(A,P, Ra, Rb, B, Text2);
27          recv_3(P,A, Text5, {Ra,Kab,B,Text4}k(A,P), T );
28          claim(A,Running,B,Kab,Text6);
29          send_4(A,B, Text7, T, {Rpa,Rb,Text6}Kab );
30          recv_5(B,A, Text9, {Rb,Rpa,Text8}Kab );
31
32          claim(A,Commit,B,Kab,Text6,Text8);
33          claim(A,Secret,Kab);
```

```
34          claim(A,Secret,Text6);
35          claim(A,Secret,Text8);
36          claim(A,Alive);
37          claim(A,Weakagree);
38      }
39      role B
40      {
41          fresh Rb: Nonce;
42          var Rpa: Nonce;
43          var Kab: SessionKey;
44          fresh Text1,Text8,Text9: Ticket;
45          var Text3,Text6,Text7: Ticket;
46
47          send_1(B,A, Rb, Text1);
48          recv_4(A,B, Text7, {Rb,Kab,A,Text3}k(B,P), {Rpa,Rb,Text6}Kab
                );
49          claim(B,Running,A,Kab,Text6,Text8);
50          send_5(B,A, Text9, {Rb,Rpa,Text8}Kab );
51
52          claim(B,Commit,A,Kab,Text6);
53          claim(B,Secret,Kab);
54          claim(B,Secret,Text6);
55          claim(B,Secret,Text8);
56          claim(B,Alive);
57          claim(B,Weakagree);
58      }
59      role P
60      {
61          var Ra, Rb: Nonce;
62          fresh Kab: SessionKey;
63          fresh Text3,Text4,Text5: Ticket;
64          var Text2: Ticket;
65
66          recv_2(A,P, Ra, Rb, B, Text2);
67          send_3(P,A, Text5, {Ra,Kab,B,Text4}k(A,P),
68                  {Rb,Kab,A,Text3}k(B,P) );
69      }
70  }
71
72  protocol @keysymm26(A,B,P)
73  {
74      role A
75      {
76          var TVPN: Nonce;
77          var Kab: SessionKey;
78          var Text: Ticket;
79
80          recv_!1(B,A, { TVPN, Kab, B, Text }k(P,A) );
81          send_!2(A,B, { TVPN, Kab, B, Text }k(A,P) );
82      }
83      role B
84      {
85      }
86      role P
87      {
88      }
89  }
```

## B.2 ISO/IEC 9798-3 protocols

Listing 11: ISO/IEC 9798-3-1

```
/*
 * Modeled from ISO/IEC 9798
 * Modeler: Cas Cremers, Dec. 2010
 *
 * signature
 * one-pass
 * unilateral
 */
const Cert: Function;

protocol isoiec-9798-3-1(A,B)
{
   role A
   {
      fresh TNA: Nonce;
      fresh Text1,Text2: Ticket;

      claim(A,Running,B,TNA,Text1);
      send_1(A,B, Cert(A),TNA,B,Text2, { TNA, B, Text1 }sk(A) );
   }
   role B
   {
      var TNA: Nonce;
      var Text1,Text2: Ticket;

      recv_1(A,B, Cert(A),TNA,B,Text2, { TNA, B, Text1 }sk(A) );

      claim(B,Commit,A,TNA,Text1);
      claim(B,Alive);
      claim(B,Weakagree);
   }
}
```

Listing 12: ISO/IEC 9798-3-2

```
/*
 * Modeled from ISO/IEC 9798
 * Modeler: Cas Cremers, Dec. 2010
 *
 * signature
 * two-pass
 * unilateral
 */
const Cert: Function;

protocol isoiec-9798-3-2(A,B)
{
   role A
   {
      var Rb: Nonce;
      fresh Ra: Nonce;
      var Text1: Ticket;
      fresh Text2,Text3: Ticket;
```

```
19
20          recv_1(B,A, Rb,Text1 );
21          claim(A,Running,B,Ra,Rb,Text2);
22          send_2(A,B, Cert(A),Ra,Rb,B,Text3, { Ra, Rb, B, Text2 }sk(A)
                );
23      }
24      role B
25      {
26          fresh Rb: Nonce;
27          var Ra: Nonce;
28          fresh Text1: Ticket;
29          var Text2,Text3: Ticket;
30
31          send_1(B,A, Rb,Text1 );
32          recv_2(A,B, Cert(A),Ra,Rb,B,Text3, { Ra, Rb, B, Text2 }sk(A)
                );
33
34          claim(B,Commit,A,Ra,Rb,Text2);
35          claim(B,Alive);
36          claim(B,Weakagree);
37      }
38  }
```

Listing 13: ISO/IEC 9798-3-3

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * signature
6    * two-pass
7    * mutual
8    */
9   const Cert: Function;
10
11  protocol isoiec-9798-3-3(A,B)
12  {
13      role A
14      {
15          fresh TNA: Nonce;
16          var TNB: Nonce;
17          fresh Text1,Text2: Ticket;
18          var Text3,Text4: Ticket;
19
20          claim(A,Running,B,TNA,Text1);
21          send_1(A,B, Cert(A), TNA, B,Text2, { TNA, B, Text1 }sk(A) );
22          recv_2(B,A, Cert(B), TNB, A,Text4, { TNB, A, Text3 }sk(B) );
23
24          claim(A,Commit,B,TNB,Text3);
25          claim(A,Alive);
26          claim(A,Weakagree);
27      }
28      role B
29      {
30          var TNA: Nonce;
31          fresh TNB: Nonce;
32          var Text1,Text2: Ticket;
```

```
33        fresh Text3,Text4: Ticket;
34
35        recv_1(A,B, Cert(A), TNA, B,Text2, { TNA, B, Text1 }sk(A) );
36        claim(B,Running,A,TNB,Text3);
37        send_2(B,A, Cert(B), TNB, A,Text4, { TNB, A, Text3 }sk(B) );
38
39        claim(B,Commit,A,TNA,Text1);
40        claim(B,Alive);
41        claim(B,Weakagree);
42     }
43  }
```

Listing 14: ISO/IEC 9798-3-4

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * signature
6    * three-pass
7    * mutual
8    */
9   const Cert: Function;
10
11  protocol isoiec-9798-3-4(A,B)
12  {
13     role A
14     {
15        var RB: Nonce;
16        fresh RA: Nonce;
17        var Text1,Text4,Text5: Ticket;
18        fresh Text2,Text3: Ticket;
19
20        recv_1(B,A, RB,Text1 );
21        claim(A,Running,B,RA,RB,Text2);
22        send_2(A,B, Cert(B), RA,RB,B,Text3, { RA, RB, B, Text2 }sk(A)
              );
23        recv_3(B,A, Cert(A), RB,RA,A,Text5, { RB, RA, A, Text4 }sk(B)
              );
24
25        claim(A,Commit,B,RA,RB,Text2,Text4);
26        claim(A,Alive);
27        claim(A,Weakagree);
28     }
29     role B
30     {
31        fresh RB: Nonce;
32        var RA: Nonce;
33        fresh Text1,Text4,Text5: Ticket;
34        var Text2,Text3: Ticket;
35
36        send_1(B,A, RB,Text1 );
37        recv_2(A,B, Cert(B), RA,RB,B,Text3, { RA, RB, B, Text2 }sk(A)
              );
38        claim(B,Running,A,RA,RB,Text2,Text4);
39        send_3(B,A, Cert(A), RB,RA,A,Text5, { RB, RA, A, Text4 }sk(B)
              );
```

```
40
41          claim(B,Commit,A,RA,RB,Text2);
42          claim(B,Alive);
43          claim(B,Weakagree);
44      }
45  }
```

Listing 15: ISO/IEC 9798-3-5

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * signature
6    * two-pass
7    * mutual
8    * parallel
9    */
10  const Cert: Function;
11
12  protocol isoiec-9798-3-5(A,B)
13  {
14      role A
15      {
16          fresh RA: Nonce;
17          var RB: Nonce;
18          fresh Text1,Text3,Text4: Ticket;
19          var Text2,Text5,Text6: Ticket;
20
21          send_1(A,B, Cert(A), RA,Text1 );
22          recv_2(B,A, Cert(B), RB,Text2 );
23          recv_3(B,A, RB,RA,A,Text6, { RB, RA, A, Text5 }sk(B) );
24          claim(A,Running,B,RA,RB,Text3,Text5);
25          send_4(A,B, RA,RB,B,Text4, { RA, RB, B, Text3 }sk(A) );
26
27          claim(A,Commit,B,RA,RB,Text5);
28          claim(A,Alive);
29          claim(A,Weakagree);
30      }
31      role B
32      {
33          var RA: Nonce;
34          fresh RB: Nonce;
35          var Text1,Text3,Text4: Ticket;
36          fresh Text2,Text5,Text6: Ticket;
37
38          recv_1(A,B, Cert(A), RA,Text1 );
39          send_2(B,A, Cert(B), RB,Text2 );
40          claim(B,Running,A,RA,RB,Text5);
41          send_3(B,A, RB,RA,A,Text6, { RB, RA, A, Text5 }sk(B) );
42          recv_4(A,B, RA,RB,B,Text4, { RA, RB, B, Text3 }sk(A) );
43
44          claim(B,Commit,A,RA,RB,Text3,Text5);
45          claim(B,Alive);
46          claim(B,Weakagree);
47      }
48  }
```

Listing 16: ISO/IEC 9798-3-6-1

```
1
2
3  /*
4   * Modeled from ISO standard
5   *
6   * signature
7   * ttp
8   * five-pass
9   * mutual
10  *
11  * A initiates and also communicates with T
12  *
13  * parameters:
14  *
15  *     NAME
16  *     IA
17  *     IB
18  *     ResA
19  *     ResB
20  *     TokenAB
21  *     TokenBA (although identical in both cases)
22  *     TokenTA
23  *
24  */
25 protocol isoiec-9798-3-6-1(A,B,T)
26 {
27  role A
28  {
29   fresh Ra,Rpa: Nonce;
30   fresh Text1,Text4,Text8,Text9: Ticket;
31   var Rb: Nonce;
32   var Text2,Text3;
33   var Text5,Text6,Text7: Ticket;
34
35   send_1(A,B, Ra,A,Text1);
36   recv_2(B,A, B,Ra,Rb,Text3,{B,Ra,Rb,A,Text2}sk(B));
37   send_3(A,T, Rpa,Rb,A,B,Text4);
38   recv_4(T,A, Text7,A,pk(A),B,pk(B),{Rpa,B,pk(B),Text6}sk(T),{Rb,A,
         pk(A),Text5}sk(T));
39   claim(A,Running,B,Ra,Rb,Text8);
40   send_5(A,B, Text9,A,pk(A),{Rb,A,pk(A),Text5}sk(T),{Rb,Ra,B,A,
         Text8}sk(A));
41
42   claim(A,Commit,B,Ra,Rb,Text2);
43   claim(A,Alive);
44  }
45  role B
46  {
47   var Ra,Rpa: Nonce;
48   var Text1,Text5,Text8,Text9: Ticket;
49   fresh Text2,Text3,Text4: Ticket;
50   fresh Rb: Nonce;
51
52   recv_1(A,B, Ra,A,Text1);
53   claim(B,Running,A,Ra,Rb,Text2);
54   send_2(B,A, B,Ra,Rb,Text3,{B,Ra,Rb,A,Text2}sk(B));
```

```
55    recv_5(A,B, Text9,A,pk(A),{Rb,A,pk(A),Text5}sk(T),{Rb,Ra,B,A,
          Text8}sk(A));
56
57    claim(B,Commit,A,Ra,Rb,Text8);
58    claim(B,Alive);
59   }
60   role T
61   {
62    var Rpa, Rb: Nonce;
63    var Text4: Ticket;
64    fresh Text5,Text6,Text7: Ticket;
65
66    recv_3(A,T, Rpa,Rb,A,B,Text4);
67    send_4(T,A, Text7,A,pk(A),B,pk(B),{Rpa,B,pk(B),Text6}sk(T),{Rb,A,
          pk(A),Text5}sk(T));
68   }
69  }
```

Listing 17: ISO/IEC 9798-3-6-2

```
1
2
3   /*
4    * Modeled from ISO standard
5    *
6    * signature
7    * ttp
8    * five-pass
9    * mutual
10   *
11   * A initiates and also communicates with T
12   *
13   * parameters:
14   *
15   *     NAME
16   *     IA
17   *     IB
18   *     ResA
19   *     ResB
20   *     TokenAB
21   *     TokenBA (although identical in both cases)
22   *     TokenTA
23   *
24   */
25  protocol isoiec-9798-3-6-2(A,B,T)
26  {
27   role A
28   {
29    fresh Ra,Rpa: Nonce;
30    fresh Text1,Text4,Text8,Text9: Ticket;
31    var Rb: Nonce;
32    var Text2,Text3;
33    var Text5,Text6,Text7: Ticket;
34
35    send_1(A,B, Ra,A,Text1);
36    recv_2(B,A, B,Ra,Rb,Text3,{B,Ra,Rb,A,Text2}sk(B));
37    send_3(A,T, Rpa,Rb,A,B,Text4);
```

```
38    recv_4(T,A, Text7,A,pk(A),B,pk(B),{Rpa,Rb,A,pk(A),B,pk(B),Text5}
          sk(T));
39    claim(A,Running,B,Ra,Rb,Text8);
40    send_5(A,B, Rpa,Text9,A,pk(A),B,pk(B),{Rpa,Rb,A,pk(A),B,pk(B),
          Text5}sk(T),{Rb,Ra,B,A,Text8}sk(A));
41
42    claim(A,Commit,B,Ra,Rb,Text2);
43    claim(A,Alive);
44   }
45   role B
46   {
47    var Ra,Rpa: Nonce;
48    var Text1,Text5,Text8,Text9: Ticket;
49    fresh Text2,Text3,Text4: Ticket;
50    fresh Rb: Nonce;
51
52    recv_1(A,B, Ra,A,Text1);
53    claim(B,Running,A,Ra,Rb,Text2);
54    send_2(B,A, B,Ra,Rb,Text3,{B,Ra,Rb,A,Text2}sk(B));
55    recv_5(A,B, Rpa,Text9,A,pk(A),B,pk(B),{Rpa,Rb,A,pk(A),B,pk(B),
          Text5}sk(T),{Rb,Ra,B,A,Text8}sk(A));
56
57    claim(B,Commit,A,Ra,Rb,Text8);
58    claim(B,Alive);
59   }
60   role T
61   {
62    var Rpa, Rb: Nonce;
63    var Text4: Ticket;
64    fresh Text5,Text6,Text7: Ticket;
65
66    recv_3(A,T, Rpa,Rb,A,B,Text4);
67    send_4(T,A, Text7,A,pk(A),B,pk(B),{Rpa,Rb,A,pk(A),B,pk(B),Text5}
          sk(T));
68   }
69  }
```

Listing 18: ISO/IEC 9798-3-7-1

```
1   /*
2    * Modeled from ISO standard
3    *
4    * signature
5    * ttp
6    * five-pass
7    * mutual
8    *
9    * B initiates and A communicates with T
10   *
11   * parameters:
12   *
13   *     NAME
14   *     IA
15   *     IB
16   *     ResA
17   *     ResB
18   *     TokenAB
```

```
 *    TokenBA (although identical in both cases)
 *    TokenTA
 *
 */
protocol isoiec-9798-3-7-1(A,B,T)
{
 role A
 {
  fresh Ra,Rpa: Nonce;
  var Rb: Nonce;
  var Text1,Text3,Text4,Text5,Text8,Text9: Ticket;
  fresh Text2,Text6,Text7: Ticket;

  recv_1(B,A, Rb,B,Text1 );
  send_2(A,T, Rpa,Rb,A,Text2 );
  recv_3(T,A, Text5, A,pk(A),B,pk(B),{Rpa,B,pk(B),Text4}sk(T),{Rb,A
      ,pk(A),Text3}sk(T) );
  claim(A,Running,B,Ra,Rb,Text6);
  send_4(A,B, A, Text7,Ra,A,pk(A),{Rb,A,pk(A),Text3}sk(T),{Rb,Ra,B,
      A,Text6}sk(A) );
  recv_5(B,A, Ra,Rb,Text9,{A,Ra,Rb,B,Text8}sk(B) );

  claim(A,Commit,B,Ra,Rb,Text8);
  claim(A,Alive);
 }
 role B
 {
  fresh Text1,Text8,Text9: Ticket;
  fresh Rb: Nonce;
  var Text3,Text4,Text6,Text7: Ticket;
  var Ra,Rpa: Nonce;

  send_1(B,A, Rb,B,Text1 );
  recv_4(A,B, A, Text7,Ra,A,pk(A),{Rb,A,pk(A),Text3}sk(T),{Rb,Ra,B,
      A,Text6}sk(A) );
  claim(B,Running,A,Ra,Rb,Text8);
  send_5(B,A, Ra,Rb,Text9,{A,Ra,Rb,B,Text8}sk(B) );

  claim(B,Commit,A,Ra,Rb,Text6);
  claim(B,Alive);
 }
 role T
 {
  var Rpa,Rb: Nonce;
  var Text2: Ticket;
  fresh Text3,Text4,Text5: Ticket;

  recv_2(A,T, Rpa,Rb,A,Text2 );
  send_3(T,A, Text5, A,pk(A),B,pk(B),{Rpa,B,pk(B),Text4}sk(T),{Rb,A
      ,pk(A),Text3}sk(T) );
 }
}
```

Listing 19: ISO/IEC 9798-3-7-2

```

```

```
3  /*
4   * Modeled from ISO standard
5   *
6   * signature
7   * ttp
8   * five-pass
9   * mutual
10  *
11  * B initiates and A communicates with T
12  *
13  * parameters:
14  *
15  *     NAME
16  *     IA
17  *     IB
18  *     ResA
19  *     ResB
20  *     TokenAB
21  *     TokenBA (although identical in both cases)
22  *     TokenTA
23  *
24  */
25 protocol isoiec-9798-3-7-2(A,B,T)
26 {
27  role A
28  {
29   fresh Ra,Rpa: Nonce;
30   var Rb: Nonce;
31   var Text1,Text3,Text4,Text5,Text8,Text9: Ticket;
32   fresh Text2,Text6,Text7: Ticket;
33
34   recv_1(B,A, Rb,B,Text1 );
35   send_2(A,T, Rpa,Rb,A,Text2 );
36   recv_3(T,A, Text5, A,pk(A),B,pk(B),{Rpa,Rb,A,pk(A),B,pk(B),Text3}
         sk(T) );
37   claim(A,Running,B,Ra,Rb,Text6);
38   send_4(A,B, A, Rpa,Text7,A,pk(A),B,pk(B),{Rpa,Rb,A,pk(A),B,pk(B),
         Text3}sk(T),{Rb,Ra,B,A,Text6}sk(A) );
39   recv_5(B,A, Ra,Rb,Text9,{Ra,Rb,A,B,Text8}sk(B) );
40
41   claim(A,Commit,B,Ra,Rb,Text8);
42   claim(A,Alive);
43  }
44  role B
45  {
46   fresh Text1,Text8,Text9: Ticket;
47   fresh Rb: Nonce;
48   var Text3,Text4,Text6,Text7: Ticket;
49   var Ra,Rpa: Nonce;
50
51   send_1(B,A, Rb,B,Text1 );
52   recv_4(A,B, A, Rpa,Text7,A,pk(A),B,pk(B),{Rpa,Rb,A,pk(A),B,pk(B),
         Text3}sk(T),{Rb,Ra,B,A,Text6}sk(A) );
53   claim(B,Running,A,Ra,Rb,Text8);
54   send_5(B,A, Ra,Rb,Text9,{Ra,Rb,A,B,Text8}sk(B) );
55
56   claim(B,Commit,A,Ra,Rb,Text6);
```

```
57    claim(B,Alive);
58   }
59   role T
60   {
61    var Rpa,Rb: Nonce;
62    var Text2: Ticket;
63    fresh Text3,Text4,Text5: Ticket;
64
65    recv_2(A,T, Rpa,Rb,A,Text2 );
66    send_3(T,A, Text5, A,pk(A),B,pk(B),{Rpa,Rb,A,pk(A),B,pk(B),Text3}
          sk(T) );
67   }
68  }
```

## B.3 ISO/IEC 9798-4 protocols

Listing 20: ISO/IEC 9798-4-1

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010, Feb. 2011.
4    *
5    * History:
6    *
7    * - v2.0, Feb. 2011:
8    *   Added key symmetry emulation protocol.
9    *
10   * ccf
11   * one-pass
12   * unilateral
13   *
14   * The identifier B is optional and may be omitted if the key is
       unidirectional.
15   *
16   * Modeling notes:
17   *
18   * - The keyed CCF (f_kab(x)) is modeled as f(x,kab)
19   */
20  hashfunction f;
21
22  protocol @keysymm-41(A,B)
23  {
24     role A
25     {
26        var X,Y,Z: Ticket;
27
28        recv_!1(B,A, f(X,Y,Z, k(A,B) ) );
29        send_!2(A,B, f(X,Y,Z, k(B,A) ) );
30     }
31     role B
32     {
33     }
34  }
35
36  protocol isoiec-9798-4-1(A,B)
```

```
{
   role A
   {
      fresh Text1,Text2: Ticket;
      fresh TNA: Nonce;

      claim(A,Running,B,TNA,Text1);
      send_1(A,B, TNA, Text2, f( TNA, B, Text1 ,k(A,B) ) );
   }
   role B
   {
      var TNA: Nonce;
      var Text1,Text2: Ticket;

      recv_1(A,B, TNA, Text2, f( TNA, B, Text1 ,k(A,B) ) );

      claim(B,Commit,A,TNA,Text1);
      claim(B,Alive);
      claim(B,Weakagree);
   }
}
```

Listing 21: ISO/IEC 9798-4-1 with unidirectional keys

```
/*
 * Modeled from ISO/IEC 9798
 * Modeler: Cas Cremers, Dec. 2010
 *
 * ccf
 * one-pass
 * unilateral
 *
 * Unidirectional key version.
 *
 * Modeling notes:
 *
 * - The keyed CCF (f_kab(x)) is modeled as f(x,kab)
 */
hashfunction f;

protocol isoiec-9798-4-1-udkey(A,B)
{
   role A
   {
      fresh Text1,Text2: Ticket;
      fresh TNA: Nonce;

      claim(A,Running,B,TNA,Text1);
      send_1(A,B, TNA, Text2, f( TNA, Text1 ,k(A,B) ) );
   }
   role B
   {
      var TNA: Nonce;
      var Text1,Text2: Ticket;

      recv_1(A,B, TNA, Text2, f( TNA, Text1 ,k(A,B) ) );

```

```
34        claim(B,Commit,A,TNA,Text1);
35        claim(B,Alive);
36        claim(B,Weakagree);
37     }
38  }
```

Listing 22: ISO/IEC 9798-4-2

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010, Feb. 2011.
4    *
5    * History:
6    *
7    * - v2.0, Feb. 2011:
8    *   Added key symmetry emulation protocol.
9    *
10   * ccf
11   * unilateral
12   * two-pass
13   *
14   * The identifier B is optional and may be omitted if the key is
       unidirectional.
15   *
16   * Modeling notes:
17   *
18   * - The keyed CCF (f_kab(x)) is modeled as f(x,kab)
19   */
20  hashfunction f;
21
22  protocol @keysymm-42(A,B)
23  {
24     role A
25     {
26        var X,Y,Z: Ticket;
27
28        recv_!1(B,A, f(X,Y,Z, k(A,B) ) );
29        send_!2(A,B, f(X,Y,Z, k(B,A) ) );
30     }
31     role B
32     {
33     }
34  }
35
36  protocol isoiec-9798-4-2(A,B)
37  {
38     role A
39     {
40        var Rb: Nonce;
41        var Text1: Ticket;
42        fresh Text2,Text3: Ticket;
43
44        recv_1(B,A, Rb,Text1 );
45        claim(A,Running,B,Rb,Text2);
46        send_2(A,B, Text3, f( Rb, B, Text2, k(A,B)) );
47     }
48     role B
```

```
49      {
50          fresh Rb: Nonce;
51          fresh Text1: Ticket;
52          var Text2,Text3: Ticket;
53
54          send_1(B,A, Rb,Text1 );
55          recv_2(A,B, Text3, f( Rb, B, Text2, k(A,B)) );
56
57          claim(B,Commit,A,Rb,Text2);
58          claim(B,Alive);
59          claim(B,Weakagree);
60      }
61  }
```

Listing 23: ISO/IEC 9798-4-2 with unidirectional keys

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * ccf
6    * unilateral
7    * two-pass
8    *
9    * Unidirectional key version.
10   *
11   * Modeling notes:
12   *
13   * - The keyed CCF (f_kab(x)) is modeled as f(x,kab)
14   */
15  hashfunction f;
16
17  protocol isoiec-9798-4-2-udkey(A,B)
18  {
19      role A
20      {
21          var Rb: Nonce;
22          var Text1: Ticket;
23          fresh Text2,Text3: Ticket;
24
25          recv_1(B,A, Rb,Text1 );
26          claim(A,Running,B,Rb,Text2);
27          send_2(A,B, Text3, f( Rb, Text2, k(A,B)) );
28      }
29      role B
30      {
31          fresh Rb: Nonce;
32          fresh Text1: Ticket;
33          var Text2,Text3: Ticket;
34
35          send_1(B,A, Rb,Text1 );
36          recv_2(A,B, Text3, f( Rb, Text2, k(A,B)) );
37
38          claim(B,Commit,A,Rb,Text2);
39          claim(B,Alive);
40          claim(B,Weakagree);
41      }
```

```
}
```

Listing 24: ISO/IEC 9798-4-3

```
/*
 * Modeled from ISO/IEC 9798
 * Modeler: Cas Cremers, Dec. 2010, Feb. 2011.
 *
 * History:
 *
 * - v2.0, Feb. 2011:
 *   Added key symmetry emulation protocol.
 *
 * ccf
 * two-pass
 * mutual
 *
 * The identifiers B,A are optional and may be (independently) be
     omitted if the key is unidirectional.
 *
 * Modeling notes:
 *
 * - The keyed CCF (f_kab(x)) is modeled as f(x,kab)
 */
hashfunction f;

protocol @keysymm-43(A,B)
{
   role A
   {
      var X,Y,Z: Ticket;

      recv_!1(B,A, f(X,Y,Z, k(A,B) ) );
      send_!2(A,B, f(X,Y,Z, k(B,A) ) );
   }
   role B
   {
   }
}

protocol isoiec-9798-4-3(A,B)
{
   role A
   {
      fresh Text1,Text2: Ticket;
      var Text3,Text4: Ticket;
      fresh TNa: Nonce;
      var TNb: Nonce;

      claim(A,Running,B,TNa,Text1);
      send_1(A,B, TNa, Text2, f(TNa,B,Text1, k(A,B) ) );
      recv_2(B,A, TNb, Text4, f(TNb,A,Text3, k(A,B) ) );

      claim(A,Commit,B,TNb,Text3);
      claim(A,Alive);
      claim(A,Weakagree);
   }
```

```
53      role B
54      {
55          var TNa: Nonce;
56          fresh TNb: Nonce;
57          var Text1,Text2: Ticket;
58          fresh Text3,Text4: Ticket;
59
60          recv_1(A,B, TNa, Text2, f(TNa,B,Text1, k(A,B) ) );
61          claim(B,Running,A,TNb,Text3);
62          send_2(B,A, TNb, Text4, f(TNb,A,Text3, k(A,B) ) );
63
64          claim(B,Commit,A,TNa,Text1);
65          claim(B,Alive);
66          claim(B,Weakagree);
67      }
68  }
```

Listing 25: ISO/IEC 9798-4-3 with unidirectional keys

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * ccf
6    * two-pass
7    * mutual
8    *
9    * Unidirectional key version.
10   *
11   * Modeling notes:
12   *
13   * - The keyed CCF (f_kab(x)) is modeled as f(x,kab)
14   */
15  hashfunction f;
16
17  protocol isoiec-9798-4-3-udkey(A,B)
18  {
19      role A
20      {
21          fresh Text1,Text2: Ticket;
22          var Text3,Text4: Ticket;
23          fresh TNa: Nonce;
24          var TNb: Nonce;
25
26          claim(A,Running,B,TNa,Text1);
27          send_1(A,B, TNa, Text2, f(TNa,Text1, k(A,B) ) );
28          recv_2(B,A, TNb, Text4, f(TNb,Text3, k(B,A) ) );
29
30          claim(A,Commit,B,TNb,Text3);
31          claim(A,Alive);
32          claim(A,Weakagree);
33      }
34      role B
35      {
36          var TNa: Nonce;
37          fresh TNb: Nonce;
38          var Text1,Text2: Ticket;
```

```
39        fresh Text3,Text4: Ticket;
40
41        recv_1(A,B, TNa, Text2, f(TNa,Text1, k(A,B) ) );
42        claim(B,Running,A,TNb,Text3);
43        send_2(B,A, TNb, Text4, f(TNb,Text3, k(B,A) ) );
44
45        claim(B,Commit,A,TNa,Text1);
46        claim(B,Alive);
47        claim(B,Weakagree);
48     }
49  }
```

Listing 26: ISO/IEC 9798-4-4

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010, Feb. 2011.
4    *
5    * History:
6    *
7    * - v2.0, Feb. 2011:
8    *   Added key symmetry emulation protocol.
9    *
10   * ccf
11   * mutual
12   * three-pass
13   *
14   * The identifier B is optional and may be omitted if the key is
         unidirectional.
15   *
16   * Modeling notes:
17   *
18   * - The keyed CCF (f_kab(x)) is modeled as f(x,kab)
19   */
20  hashfunction f;
21
22  protocol @keysymm-44(A,B)
23  {
24     role A
25     {
26        var X,Y,Z: Ticket;
27
28        recv_!1(B,A, f(X,Y,Z, k(A,B) ) );
29        send_!2(A,B, f(X,Y,Z, k(B,A) ) );
30     }
31     role B
32     {
33        var X,Y,Z,ZZ: Ticket;
34
35        recv_!3(A,B, f(X,Y,Z,ZZ, k(A,B) ) );
36        send_!4(B,A, f(X,Y,Z,ZZ, k(B,A) ) );
37     }
38  }
39
40  protocol isoiec-9798-4-4(A,B)
41  {
42     role A
```

```
43      {
44          fresh Ra: Nonce;
45          var Rb: Nonce;
46          var Text1,Text4,Text5: Ticket;
47          fresh Text2,Text3: Ticket;
48
49          recv_1(B,A, Rb, Text1 );
50          claim(A,Running,B,Ra,Rb,Text2);
51          send_2(A,B, Ra, Text3, f(Ra,Rb,B,Text2, k(A,B) ) );
52          recv_3(B,A, Text5, f(Rb,Ra,Text4, k(A,B) ) );
53
54          claim(A,Commit,B,Ra,Rb,Text2,Text4);
55          claim(A,Alive);
56          claim(A,Weakagree);
57      }
58      role B
59      {
60          var Ra: Nonce;
61          fresh Rb: Nonce;
62          fresh Text1,Text4,Text5: Ticket;
63          var Text2,Text3: Ticket;
64
65          send_1(B,A, Rb, Text1 );
66          recv_2(A,B, Ra, Text3, f(Ra,Rb,B,Text2, k(A,B) ) );
67          claim(B,Running,A,Ra,Rb,Text2,Text4);
68          send_3(B,A, Text5, f(Rb,Ra,Text4, k(A,B) ) );
69
70          claim(B,Commit,A,Ra,Rb,Text2);
71          claim(B,Alive);
72          claim(B,Weakagree);
73      }
74  }
```

Listing 27: ISO/IEC 9798-4-4 with unidirectional keys

```
1   /*
2    * Modeled from ISO/IEC 9798
3    * Modeler: Cas Cremers, Dec. 2010
4    *
5    * ccf
6    * mutual
7    * three-pass
8    *
9    * Unidirectional key version.
10   *
11   * Modeling notes:
12   *
13   * - The keyed CCF (f_kab(x)) is modeled as f(x,kab)
14   */
15  hashfunction f;
16
17  protocol isoiec-9798-4-4-udkey(A,B)
18  {
19      role A
20      {
21          fresh Ra: Nonce;
22          var Rb: Nonce;
```

```
23        var Text1,Text4,Text5: Ticket;
24        fresh Text2,Text3: Ticket;
25
26        recv_1(B,A, Rb, Text1 );
27        claim(A,Running,B,Ra,Rb,Text2);
28        send_2(A,B, Ra, Text3, f(Ra,Rb,Text2, k(A,B) ) );
29        recv_3(B,A, Text5, f(Rb,Ra,Text4, k(B,A) ) );
30
31        claim(A,Commit,B,Ra,Rb,Text2,Text4);
32        claim(A,Alive);
33        claim(A,Weakagree);
34     }
35     role B
36     {
37        var Ra: Nonce;
38        fresh Rb: Nonce;
39        fresh Text1,Text4,Text5: Ticket;
40        var Text2,Text3: Ticket;
41
42        send_1(B,A, Rb, Text1 );
43        recv_2(A,B, Ra, Text3, f(Ra,Rb,Text2, k(A,B) ) );
44        claim(B,Running,A,Ra,Rb,Text2,Text4);
45        send_3(B,A, Text5, f(Rb,Ra,Text4, k(B,A) ) );
46
47        claim(B,Commit,A,Ra,Rb,Text2);
48        claim(B,Alive);
49        claim(B,Weakagree);
50     }
51  }
```

# Listings